Multi-Objective Large Neighborhood Search

Pierre Schaus and Renaud Hartert

UCLouvain, ICTEAM,
Place sainte barbe 2,
1348 Louvain-la-Neuve, Belgium
{pierre.schaus,renaud.hartert}@uclouvain.be

Abstract. Large neighborhood search (LNS) [25] is a framework that combines the expressiveness of constraint programming with the efficiency of local search to solve combinatorial optimization problems. This paper introduces an extension of LNS, called multi-objective LNS (MO-LNS), to solve multi-objective combinatorial optimization problems ubiquitous in practice. The idea of MO-LNS is to maintain a set of nondominated solutions rather than just one best-so-far solution. At each iteration, one of these solutions is selected, relaxed and optimized in order to strictly improve the hypervolume of the maintained set of nondominated solutions. We introduce modeling abstractions into the OscaR solver for MO-LNS and show experimentally the efficiency of this approach on various multi-objective combinatorial optimization problems.

Keywords: Constraint Programming, Multi-Objective Combinatorial Optimization, Large Neighborhood Search.

Multi-Objective Combinatorial Optimization (MOCO) problems are ubiquitous in real-world applications. Decision makers often face the problem of dealing with several objectives *e.g.* the cost and the risk. In this situation, people are mostly interested to see a set of solutions representing the optimal compromises between objectives instead of one solution resulting from an *a priori* preference between these objectives.

Not surprisingly, the last decades have seen a growth of interest in the theory and the methodology for MOCO problems (see [7, 26] for a review). Currently, hybridized-meta-heuristics between Evolutionary Algorithm (EA) and Local Search (LS) obtain state-of-the-art results[1] on most standard MOCO problems such as the traveling salesman, the binary knapsack, and the quadratic assignment problems (see [1] for a review of these methods). However – despite the implementation facilities offered by libraries such as ParadisEO [4] and jMetal [6] – these approaches are quite far from "model and run" ones. Indeed, users still have to provide several implementation blocks (for crossover, mutations, moves and neighborhood, etc.) requiring a great knowledge and expertise on the problems and the used algorithms. Furthermore, meta-heuristic methods for MOCO problems are more and more specific and strongly related to the optimization problem to solve [8]. This tendency increases the difficulty to design a single universal method or solver.

[1] The LS and EA communities are probably the most active ones on the domain of MOCO.

Conversely, Constraint Programming (CP) offers a high level declarative language and has shown to be a competitive approach for solving single-objective constrained optimization problems (COP). In particular, the LNS (Large Neighborhood Search) framework [25] – which combines the efficiency of LS with the expressiveness of CP – allowed to solve large scale problems such as vehicle routing [3, 25], scheduling [14, 21], and assignment/bin-packing problems [18, 23] successfully.

We believe that the expressiveness of CP can have a real added value to tackle some MOCO problems by reducing the amount of work required from the modeler.[2] This work is one step in the direction of extending the LNS framework in the multi-objective context (MO-LNS). The goal of MO-LNS is to quickly discover good nondominated sets of solutions for large scale MOCO problems while keeping a declarative CP model.

This paper introduces the MO-LNS framework. We demonstrate experimentally its flexibility on standard MOCO problems as well as on a real-world bi-objective version of the Tank Allocation Problem (TAP) [24]. We also introduce modeling abstractions, explaining in depth an MO-LNS model implemented with the OscaR open source library [20].

Outline. Section 1 gives definitions related to constraint programming and multi-objective optimization. Section 2 reviews the related work of existing CP approaches to solve MOCO problems. Section 3 introduces MO-LNS. Section 4 details an MO-LNS model for the quadratic assignment problem in the OscaR [20] solver. Section 5 experiments the MO-LNS approach on various MOCO problems. Section 6 gives perspectives and concludes.

1 Definitions

The typical MOCO problem we want to solve has m integer objective variables to minimize while satisfying some constraints:

$$\begin{array}{ll} \text{Minimize} & obj = (obj_1, obj_2, \ldots, obj_m) \\ \text{Subject to} & constraints \end{array} \tag{1}$$

Solutions of this problem are defined as follows:

Definition 1 (Solution). *Let $\mathcal{P}$ be a MOCO problem, a solution of the problem $\mathcal{P}$ is an assignment of the decision variables and objective variables of $\mathcal{P}$ that satisfies all the constraint of this problem. In the following, $sol(x)$ denotes the value assigned to the variable x in the solution sol.*

The conflicting nature of the objectives usually prevents the existence of a unique solution sol^* that is optimal in all objectives. Hence, one is usually interested in the set of all the optimal compromises known as *Pareto optimal* solutions.

[2] The lack of hybridization with CP approaches for solving MOCO problems was recently underlined by Ehrgott in [8].

Definition 2 (Pareto dominance). *Let sol and sol' be two solutions of a MOCO problem $\mathcal{P}$. We say that sol dominates sol', denoted $sol \prec sol'$, if and only if:*

$$\begin{aligned} & \forall j \in [1..m] : \ sol(obj_j) \leq sol'(obj_j) \\ \wedge \quad & \exists j \in [1..m] : \ sol(obj_j) < sol'(obj_j) \end{aligned} \tag{2}$$

Besides, we say that sol weakly-dominates sol', denoted $sol \preceq sol'$, if and only if the first part of Equation 2 holds.[3]

Definition 3 (Pareto optimality). *Let $sols(\mathcal{P})$ denotes all the feasible solutions of a MOCO problem $\mathcal{P}$. A solution sol^* is Pareto optimal if and only if there is no solution sol' in $sols(\mathcal{P})$ that dominates sol^*:*

$$\nexists sol' \in sols(\mathcal{P}) : sol' \prec sol^* \tag{3}$$

In other words, a solution is said to be Pareto optimal if it is impossible to improve the value of one objective without degrading the value of at least one other objective.

The set of all the Pareto optimal solutions is known as the *Pareto set* and is defined as follows:

Definition 4 (Pareto set). *The Pareto set of a MOCO problem $\mathcal{P}$ is the set of all the Pareto optimal solutions of this problem:*

$$\{sol \in sols(\mathcal{P}) \mid \nexists sol' \in sols(\mathcal{P}) : sol' \prec sol\} \tag{4}$$

Definition 5 (Pareto front). *The Pareto front of a MOCO problem $\mathcal{P}$ is the projection of its Pareto set in the objective space.*

Unfortunately, discovering the exact Pareto set may be impracticable on difficult MOCO problems. We are thus interested in finding an approximation of this set, also known as the *archive*.

Definition 6 (Archive). *An archive $\mathcal{A}$ is a set of solutions such that there is no solution in the archive that dominates an other solution in the archive. This property is known as the domination-free property:*

$$\forall sol \in \mathcal{A}, \nexists sol' \in \mathcal{A} : sol' \prec sol \tag{5}$$

As illustrated in Fig. 1, an archive can be used to partition the objective space into three subspaces:

- The *dominated subspace* consists of all the solutions that are dominated by at least one solution in the archive (see Fig. 1a);
- The *diversification subspace* consists of all the solutions that neither dominate nor are dominated by any solution in the archive (see Fig. 1b);
- The *intensification subspace* consists of all the solutions that dominate at least one solution in the archive (see Fig. 1c).

Clearly the archive quality can only be improved by adding new solutions from:

[3] In the remainder of this paper, we abuse of these notations to compare solutions with vectors.

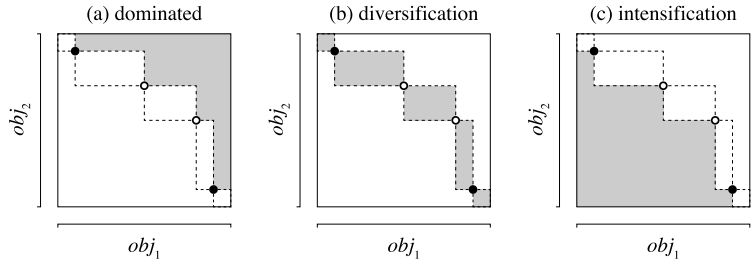

Fig. 1. An archive partitions the objective space into three subspaces: (a) the dominated subspace, (b) the diversification subspace, and (c) the intensification subspace.

- the intensification subspace where a new solution replaces at least one solution in the archive;
- the diversification subspace where a new solution is added into the archive without replacing any other solutions.

In the following, we suppose that an archive maintains its domination-free property by removing the solutions that are dominated by a new solution from the intensification space. Therefore, adding new solutions into the archive increases the size of the dominated subspace. The size of the dominated subspace is a common indicator used to measure the quality of an archive known as the hypervolume indicator $\mathcal{H}$ [30]:

Definition 7 (Hyper-volume indicator). *The hypervolume $\mathcal{H}$ is an unary quality indicator (to be maximized) which measures the volume of the objective subspace dominated by a given archive.*

The hypervolume indicator is mostly used for bi-objective problems since its computation increases exponentially with the number of objectives.

Every solution of the Pareto front is not equally difficult to discover. Supported solutions can be discovered using a single objective optimization approach by minimizing a linear aggregation of the objectives while non supported ones cannot [7]:

Definition 8 (Supported Pareto optimal solutions). *A supported Pareto optimal solution is an extreme point on the convex hull of the Pareto front.*

The Pareto front has no guarantee to be convex, justifying the need for more advanced techniques to tackle MOCO problems.

2 Related work

While multi-objective combinatorial optimization problems have gained a lot of traction over last decade in the Local Search and Evolutionary Search communities (with algorithms such as NSGA-II [5] and SPEA-II [29]), not so many methods have been proposed for CP.

One approach detailed in Section 2.1 has been initially proposed to solve bi-objective problems by solving a sequence of problems. Another approach detailed in Section 2.2

allows to solve arbitrary multi-objective problems in one search using an adaptation of Branch and Bound (BnB) search with a special global constraint to filter the objective variables.

2.1 Bi-Objective Optimization

In bi-objective optimization problems, improving the value of the first objective of a Pareto optimal solution cannot be done without degrading the value of the second objective. The approach proposed by van Wassenhove and Gelders [27] exploits this property in order to find the exact Pareto optimal set of solutions of bi-objective optimization problems. The idea is as follows:[4]

1. Find the Pareto optimal solution with the best value for the first objective;
2. If this solution exists, the search is restarted with an additional constraint enforcing the value of the second objective to be strictly better than its value in the previous solution.

2.2 Multiple-Objective Optimization with CP (MO-CP)

In [9], Gavanelli suggested a framework to solve multi-objective optimization with CP allowing to find all the Pareto optimal solutions in a single search. This framework is presented as a specialized BnB search making use of no-goods recording, corresponding to nondominated solutions. Although not presented this way in [9], we view this approach as the introduction of a new global constraint defined on the objective variables and an archive $\mathcal{A}$ that is domination-free:

$$\texttt{Pareto}(obj_1, \ldots, obj_m, \mathcal{A} = \{sol_1, \ldots, sol_n\}) \tag{6}$$

where sol_i is a solution to Problem (1). The `Pareto` constraint ensures that the next discovered solution is nondominated w.r.t. $\mathcal{A}$:

$$\nexists sol \in \mathcal{A} : sol \preceq (obj_1, \ldots, obj_m) \tag{7}$$

Let $obj_i^{\min}$ and $obj_i^{\max}$ denote the lower and upper bounds of the objective variable obj_i. The filtering of $obj_i^{\max}$ achieved in [9] considers first the *dominated point* DP_i that is defined as follows:

$$DP_i = (obj_1^{\min}, \ldots, obj_{i-1}^{\min}, obj_i^{\max}, obj_{i+1}^{\min}, \ldots, obj_m^{\min}) \tag{8}$$

Then it finds a solution $sol^* \in \mathcal{A}$ dominating the dominated point *i.e.* such that $sol^* \preceq DP_i$. If such a solution exists, $sol^*(obj_i) - 1$ is an upper bound for obj_i that can be used to filter its domain:

$$obj_i^{\max} \leftarrow sol^*(obj_i) - 1. \tag{9}$$

Since we are interested in finding the tightest upper bound for objective i, the idempotent filtering rule is:

$$obj_i^{\max} \leftarrow \min(\{obj_i^{\max}\} \cup \{sol(obj_i) - 1 \mid sol \in \mathcal{A} \wedge sol \preceq DP_i\}) \tag{10}$$

[4] The approach of van Wassenhove and Gelders can be seen as a particular instance of the ϵ-constraint method [10].

In this scheme, each time a new solution is found, it is added into $\mathcal{A}$ possibly filtering out dominated solutions to maintain its domination-free property.

It has been demonstrated in [9] that MO-CP, although more general, is also more efficient than the approach of van Wassenhove and Gelders to solve bi-objective knapsack problems.[5]

Example 1. Consider $\texttt{Pareto}(obj_1, obj_2, obj_3, \mathcal{A})$ with domains $D(obj_1) = [3..5]$, $D(obj_2) = [2..5]$, $D(obj_3) = [2..5]$ and $\mathcal{A} = \{(1,4,2),(4,2,3),(2,3,1),(2,1,4)\}$. No filtering for $obj_1^{\max}$ is possible because $(obj_1^{\max} = 5, obj_2^{\min} = 2, obj_3^{\min} = 2)$ is not dominated by any point in $\mathcal{A}$. For obj_2 some filtering is possible since $(obj_1^{\min} = 3, obj_2^{\max} = 5, obj_3^{\min} = 2)$ is dominated by $(1,4,2)$ and $(2,3,1)$. We can set $obj_2^{\max} \leftarrow \min(4-1, 3-1) = 2$. The domain of obj_3 can also be filtered since $(obj_1^{\min} = 2, obj_2^{\min} = 2, obj_3^{\max} = 5)$ is dominated by $(2,1,4)$. We can thus set $obj_3^{\max} \leftarrow 4 - 1 = 3$.

3 Multi-Objective LNS

Large Neighborhood Search (LNS) [25] is an hybridization between CP and LS. At each iteration (called *restart* in the LNS context), a best-so-far solution is considered for improvement by exploration of a neighborhood using CP. This solution is relaxed and optimized again with CP, replacing the best-so-far solution on each improvement. This process is repeated until a stopping criterion is met (for instance a maximum number of restarts). LNS has the main advantage that the neighborhood to explore at each restart is potentially very large, permitting to escape local minima most of the time. Every CP optimization model can be turned easily into an LNS by providing the following information/implementation to the solver:

- A *relaxation procedure*. This procedure (also called fragment selection) defines the neighborhood to explore. It adds some constraints to the problem coming from the structure of the best-so-far solution while allowing some flexibility for re-optimization. This relaxation procedure generally includes some randomness.
- A *search limit*. This limit, although optional, prevents the search from spending too much time in the exploration of the neighborhood. It can for instance be a time limit, or a limit on the number of backtracks.

Finding the right relaxation procedure, relaxation size and search limit is a challenging problem (see [15, 16, 22] for attempts to automatize LNS parameters). This work proposes to adapt the LNS scheme in a multi-objective context.

3.1 Restarting from a nondominated solution

Instead of a unique best-so-far solution, the MO-LNS framework maintains a best-so-far approximation $\mathcal{A}$ of the Pareto set *i.e.* an *archive*. The `Pareto` constraint (using the

[5] This is probably due to the fact that the approach of Gavanelli does not need to restart the search at each discovered solution. Besides, the already discovered solutions provide some supports to prune dominated branches of the search tree.

set $\mathcal{A}$) is added to the model ensuring that only new nondominated solutions w.r.t. $\mathcal{A}$ can be discovered.

Any solution in $\mathcal{A}$ can be used as restarting point. We distinguish two kinds of improvements of the archive:

- finding a new point in the diversification subspace. We call this a *diversification* of the archive. The resulting archive has one more element;
- finding a new point in the intensification subspace. We call this an *intensification* of the archive. The resulting archive is not larger after this insertion since some points may disappear from $\mathcal{A}$.

Notice that both improvements strictly increase the hypervolume (see Definition 7) and both improvements are allowed by the `Pareto` constraint which guarantees that only nondominated solutions w.r.t. the archive can be discovered.

3.2 Guiding Diversification-Intensification

The discovery of a new solution may contribute to diversify the archive, or it may improve existing solutions. A good strategy in terms of filtering for the `Pareto` constraint could consist in finding quickly a limited number of solutions very close to the Pareto set. On the contrary, it would be less efficient to quickly discover a large number of nondominated solutions while being far from the Pareto set. Those two situations are illustrated in Fig. 2.

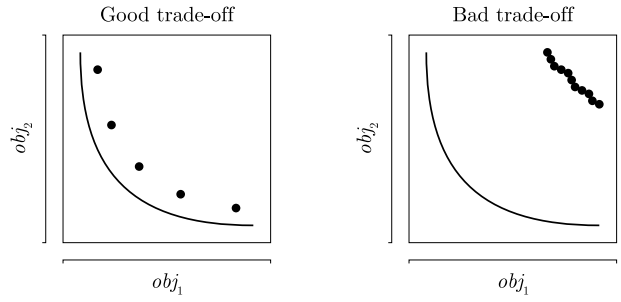

Fig. 2. Situations resulting from (left) a good diversification/intensification trade-off (right) too much diversification at the early MO-LNS iterations. The exact Pareto front is represented with the plain curve.

An archive with many solutions quite far from the Pareto set is the consequence of a too large number of diversification at the early iterations of MO-LNS. It is thus important to have a good trade-off between the number of diversification and intensification[6]. A first idea to control the ratio of diversification/intensification is to adapt the search

[6] Beck [2] also proposes to control diversification and intensification of a pool of elite solutions for single objective problems. We owe this observation to an anonymous referee. Thanks!

heuristic dynamically. One could for instance have two different search heuristics *e.g.* one that favors intensification and the other one favoring diversification. This approach has the main disadvantage of requiring a good knowledge of the problem and an additional implementation work by the modeler. A better approach forces diversification or intensification at each restart, based on a *dynamic* change of the filtering behavior of the different objectives. Each objective can be set into three different filtering modes during the BnB search:

1. *No-Filtering*: it means that the filtering of the objective is deactivated, having no impact at all.
2. *Weak-Filtering*: each time a new solution is discovered during the search, the upper bound of the objective is updated such that the next discovered solution has a lower or equal upper bound for this objective.
3. *Strong-Filtering*: each time a new solution is discovered during the search, the upper bound of the objective is updated such that the next discovered solution strictly improves the upper bound of this objective.

We propose to use this idea to control the diversification/intensification rates along the restarts.

Intensification The goal of intensification restarting from a solution sol is to discover new solutions dominating it. We propose two different ways to guarantee that the next discovered solution dominates sol by adjusting all the objective's upper bounds to their value in sol and setting the objectives in one of both following configurations:

- *Strong Intensification*. All the objectives are set in *Strong-Filtering* mode;
- *Driven Intensification*. All the objectives are set in *Weak-Filtering* mode except one that is set into *Strong-Filtering* mode. This objective drives the intensification.

Both configuration are illustrated in Fig. 3 where a possible sequence of successive discovered solutions is given.

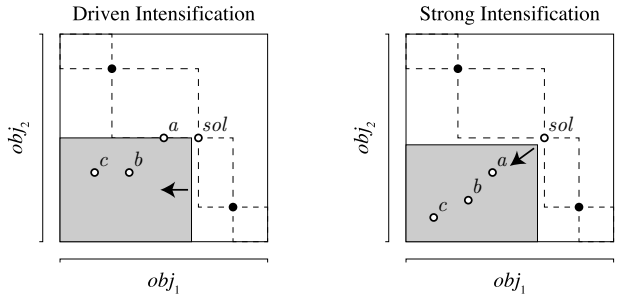

Fig. 3. Intensification. (left) obj_1 is set in *Strong-Filtering* mode and obj_2 is set in *Weak-Filtering* mode. (right) obj_1 and obj_2 are both set in *Strong-Filtering* mode. For both configurations, a possible sequence of successive discovered solutions is given.

Diversification The diversification mode attempts to find new nondominated solutions without necessarily trying to dominate existing ones. To achieve this, we set all the objectives in *No-Filtering* mode and we let the `Pareto` constraint force the discovery of new nondominated solutions.

Fig. 4 illustrates the benefit of including intensification along the restarts on a bi-objective knapsack (maximization) problem with 100 items from MOCOLib [28]. In the first setting, only diversification restarts are used. In the second setting, 50% are diversification, the others are intensification restarts. One can see on the left, that after 5 seconds, the quality of the nondominated solutions is clearly superior when using intensification restarts. On the right the evolution of the hypervolume (averaged on 10 runs) is depicted. As expected, the hypervolume grows faster when including intensification restarts.

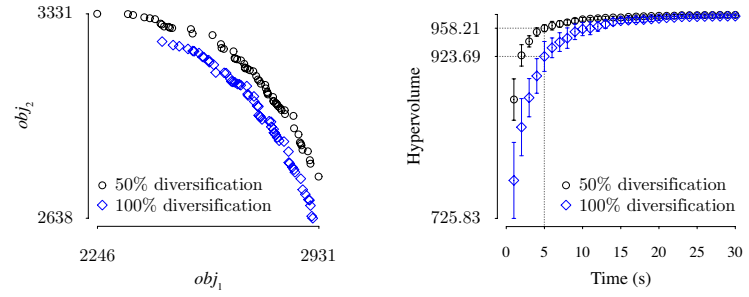

Fig. 4. Impact of the diversification/intensification ratio on a 100 items bi-objective knapsack problem. (left) Nondominated solutions obtained after 5 seconds. (right) Evolution of the hypervolume.

To summarize, the actions that must be taken at each MO-LNS restart are:

- select a solution sol from the set of nondominated solutions;
- relax sol;
- configure all objectives either in intensification or in diversification mode.

The question of selecting the nondominated solution sol is addressed next.

3.3 Selection of the restarting solution

Choosing the next solution to restart from can have a strong impact on the quality of the archive. Intuitively, a relaxed solution has a higher chance to generate new solutions close to this one in the objective space when doing diversification. We call this the *locality effect*. Having a final set of nondominated solutions spreading over the frontier is a desired property supported by many researchers [17]. A very simple idea, quite effective in practice, is to select randomly and uniformly the solution to restart from. Unfortunately, this strategy might have negative side effects caused by the locality effect. If at some point, clusters of solutions in the archive appear in the objective space,

those clusters have high chances to be reinforced. We would prefer a selection strategy helping to fill in the gaps between those clusters. We imagined another strategy, also randomized (to ensure diversification), but tending to fill in the gaps more quickly. The idea is to select an uniform random point on the hyperplane formed by the extremities of the archive (*i.e.* on a line for a bi-objective problem). The solution selected to relax is then the nearest (according to an Euclidean distance metric) one from this random point. This *nearest neighbor strategy* is illustrated in Fig. 5.

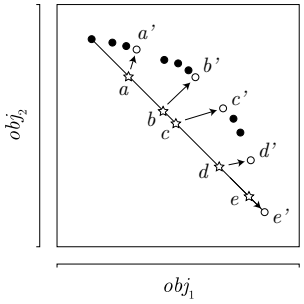

Fig. 5. Selection of solutions according to the nearest neighbor strategy. The straight line represents the hyperplane defined by the extremities of the archive. The stars a, b, c, d, and e correspond to possible points randomly generated on the hyperplane. The solutions a', b', c', d', and e' are the solutions that would be selected for each of the random points.

Fig. 6 presents the benefits of the nearest neighbor strategy over the purely randomized selection strategy on a 200 items bi-objective knapsack problem from MOCOLib [28]. We have initially added 6 Pareto optimal solutions in the archive, then 20 diversification restarts were executed with both strategies. While the pure randomized strategy (right) quickly focuses on a particular region of the objective space, the nearest neighbor strategy (left) diversifies better the objective space trying to discover solutions between the gaps on the frontier. The reason is that with the randomized nearest neighbor strategy, solutions close to the gaps are selected more frequently.

4 Modeling an MO-Quadratic Assignment Problem with MO-LNS

This section introduces the MO-LNS modeling of the Multi-Objective Quadratic Assignment Problem (MOQAP) in OscaR [20] and provides some implementation details.

In this problem a set of n facilities must be assigned to n different locations. For each pair of locations, a distance is specified and for each pair of facilities a weight or flow is specified (*e.g.* , the amount of supplies transported between the two facilities). The problem is to assign all facilities to different locations with the goal of minimizing the sum of the distances multiplied by the corresponding weights. More formally, if $x(i)$ represents the location assigned to facility i, the objective is to minimize the weighted sum $\sum_{i,j \in [1..n]} w(i,j) \cdot d(x(i), x(j))$ with w and d respectively the weight and distance matrices.

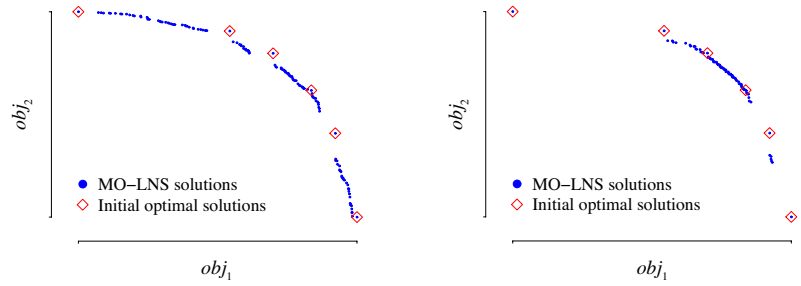

Fig. 6. Impact of selection strategy after 20 restarts on a 200 items bi-objective knapsack problem using 100% of diversification starting from 6 initial Pareto optimal solutions. (left) Using the randomized nearest neighbor strategy. (right) Using a pure randomized strategy.

The multi-objective QAP with multiple weight matrices naturally models any facility layout problem where we are concerned with the flow of more than one type of item or agent [12]. The OscaR model for a bi-objective QAP is given in Statement 1.1.

The data declaration is specified in lines 1 - 6 and should be self-explanatory. The distance matrix and the two weight matrices are declared. Then comes the CP model. A solver object is created in line 8. An array `x` of n decision variables is created at line 10 representing the location of each facility. The distance variables between any two facility are initialized at line 11 using 2D element constraints. The two objective functions are initialized in lines 12 - 13 multiplying each distance entry by the corresponding weight and summing them all.

Notice that `paretoMinimize`, `subjectTo` and `exploration` are methods of the `CPSolver` class each returning the `CPSolver` caller instance. This allows to chain directly the calls. The `paretoMinimize` method call at line 15 implicitly adds the `Pareto` global constraint (6) to the model. The search in the exploration block is a nondeterministic search [11]. Although hidden from the user point of view, all the discovered solutions are added into the archive $\mathcal{A}$ used by the `Pareto` constraint. The `run` method takes two optional arguments: a limit on the number of solutions and a limit on the number of backtracks. Both are set to infinity by default. The search to find the first feasible solution is started at line 23.

Lines 20 and 21 are iterated until all variables are bound and each iteration nondeterministically assigns a facility `x(i)` to a location `v` computed by the variable value heuristic introduced in [19]. Notice that this heuristic receives a weight matrix in argument. In the non-deterministic search `exploration` block, the weight matrix is randomly chosen between `w1` and `w2` at line 20.

The MO-LNS procedure is implemented at lines 30 - 40, after that the first feasible solution is found. The search executes 1000 LNS restarts. Each restart has a limit of 200 failures and use the search defined in the exploration block. On each restart a solution is selected from the current archive according to the nearest neighbor strategy (line 32). Then, the objectives are configured into intensification or diversification mode w.r.t. to a user defined probability. The `runSubjectTo` method is similar to the `run` method except that all the constraints added in its block are temporary constraints

that will be removed before the next restart. Some constraints are added through the `relaxRandomly` at line 38 to restore the assignments on `x` from the selected solution `sol` except for 5 randomly chosen variables.

```
1  // DATA AND CONSTANTS
2  val N = 0 until n                // number of locations
3  var w1: Array[Array[Int]] = ... // weight matrix 1
4  var w2: Array[Array[Int]] = ... // weight matrix 2
5  var d:  Array[Array[Int]] = ... // distance matrix
6  val rand = Random(0)             // random number generator
7  // CP MODEL
8  val cp = CPSolver()
9  // the location chosen for each facility
10 val x    = Array.fill(n){CPVarInt(cp, N)}
11 val dist = Array.tabulate(n, n){(i, j) => d(x(i))(x(j))}
12 val obj1 = sum(n, n){(i, j) => dist(i)(j) * w1(i)(j)}
13 val obj2 = sum(n, n){(i, j) => dist(i)(j) * w2(i)(j)}
14 // CONSTRAINT AND EXPLORATION
15 cp.paretoMinimize(obj1, obj2) subjectTo {
16   cp.add(allDifferent(x), Strong)
17 } exploration {
18   // compute variable, value heuristic randomly on w1 or w2
19   while (!allBounds(x)) {
20     val (i, v) = heuristic(if (rand.nextBoolean) w1 else w2)
21     cp.branch(cp.post(x(i) == v))(cp.post(x(i) != v))
22   }
23 } run(nbSolution = 1)     // only search for an initial solution
24 // MO LNS PARAMETERS
25 val maxRestarts    = 1000 // number of restarts
26 val maxFailures    = 200  // max number of failures at each restart
27 val relaxSize      = 5    // number of relaxed variables at each restart
28 val probaIntensify = 30   // probability (%) of intensification
29 // MO LNS FRAMEWORK
30 for (restart <- 1 to maxRestarts) {
31   // next solution to restart from
32   val sol = nearestNeibhborSol()
33   // random selection between intensification or diversification
34   if (rand.nextInt(100) < probaIntensify) cp.objective.intensify(sol)
35   else cp.objective.diversify()
36   // search
37   cp.runSubjectTo(failureLimit = maxFailures) {
38     relaxRandomly(x, sol, relaxSize)
39   }
40 }
```

Statement 1.1: Model of the multi-objective QAP in OscaR/Scala.

5 Experiments

This section compares the performances of MO-LNS over MO-CP on bi-objective problems.[7] The tested problems are: 1) the multi-objective QAP, 2) the multi-objective binary knapsack and 3) a bi-objective tank allocation problem. Although multi-objective heuristics are the methods of choice to tackle the two first problems, those are interesting standard benchmarks to study, with known exact Pareto front. Problem 3 however,

[7] Instances and optimal fronts available at `http://becool.info.ucl.ac.be/resources/mo-lns`

is more constrained and probably more suited for constraint programming. All experiments were conducted with the OscaR open-source solver [20] on an Intel® Core i7 ™ 2.6GHz CPU.

5.1 Multi-Objective Quadratic Assignment Problem

We experiment the MO-LNS model introduced in Section 4 on instances.[8] with 10 facilities from [13] Table 1 reports the results obtained with a 30 seconds timeout, averaged over 10 runs for MO-LNS. The size and the hypervolume of the exact Pareto fronts are given in columns 2 and 3. The hypervolumes obtained with MO-LNS and MO-CP ($\mathcal{H}_S$) are given in columns 4 and 5. The sizes of the archives obtained with MO-LNS and MO-CP ($|S|$) are given in columns 6 and 7. The number of optimal solutions obtained with each approach ($|S \cap S^*|$) is presented in columns 8 and 9.

As can be seen, 3 instances are optimally solved with MO-CP. MO-LNS is also able to solve these instances optimally and obtain strictly better results on the other instances. The hypervolume values reached by MO-LNS are very close to the optimal ones.

Table 1. Results on MO-QAP instances from [13] with MO-LNS and MO-CP.

			$\mathcal{H}_S(10^8)$		$\|S\|$		$\|S \cap S^*\|$	
Instance	$\|S^*\|$	$\mathcal{H}_{S^*}(10^8)$	MO-LNS	MO-CP	MO-LNS	MO-CP	MO-LNS	MO-CP
KC10-2fl-1uni	13	117.38	**115.43**	99.09	9	15	**6.6**	0
KC10-2fl-2uni	1	91.56	**87.11**	76.99	1.4	2	**0.6**	0
KC10-2fl-3uni	130	90.50	**87.42**	78.79	84.4	65	**30.6**	0
KC10-2fl-1rl	58	606005.79	**604932.58**	598146.60	54	41	**50.4**	13
KC10-2fl-2rl	15	604864.40	**604864.40**	**604864.40**	15	15	**15**	**15**
KC10-2fl-3rl	55	623898.50	**623076.50**	565772.58	47	37	**43**	0
KC10-2fl-4rl	53	732716.05	**732716.05**	**732716.05**	53	53	**53**	**53**
KC10-2fl-5rl	49	1819669.88	**1819669.88**	**1819669.88**	49	49	**49**	**49**

5.2 Multi-Objective 0/1 Knapsack Problem

The multi-objective 0/1 knapsack problem is defined as follows:

$$\begin{aligned} &\text{Maximize} \quad obj = (obj_1, \ldots, obj_m) \quad \text{with} \quad obj_j = \sum_{i=1}^{n} x_i \cdot p_{ij} \\ &\text{Subject to} \quad \sum_{i=1}^{n} x_i \cdot w_i \leq C \end{aligned} \tag{11}$$

with p_{ij} the profit of item i according to objective j, w_i the weight of item i, and C the capacity of the knapsack. The binary variables x_i represent the selection status of each item i.

MO-LNS settings The next solution sol to restart from is chosen in the archive with the nearest neighbor strategy. The idea of the relaxation procedure is to keep fixed some *good items* (w.r.t. to one objective) already selected in sol :

[8] Only the ones for which the optimal Pareto front is available.

1. select randomly one objective index $j \in [1..m]$,
2. select randomly 90% of the items in the set $\{i \in [1..n] \mid sol(x_i) = 1\}$ according to a probability function proportional to p_{ij}/w_i,
3. keep fixed the items selected at step 2. *i.e.* force them to be in the knapsack,
4. chose randomly to diversify or intensify with equal probability.

The variable-value heuristic used when re-optimizing after the relaxation is dependent of the selected objective j in the relaxation. The heuristic selects first the unbound variable `x(i)` with the largest ratio p_{ij}/w_i, selecting this item on the left and removing it on the right branch. Each restart is given a limit of 1000 backtracks.

Results We compare the MO-CP and MO-LNS approaches on instances from MOCOLib [28] ranging from 50 to 250 items. Table 2 reports the results obtained with a 60 seconds timeout, averaged over 10 runs for MO-LNS. The size and the hypervolume of the exact Pareto fronts[9] are given in columns 2 and 3. The hypervolumes obtained with MO-LNS and MO-CP ($\mathcal{H}_S$) are given in columns 4 - 5. The sizes of the archives obtained with MO-LNS and MO-CP ($|S|$) are given in columns 6 - 7. The number of optimal solutions obtained with each approach ($|S \cap S^*|$) is presented in columns 8 - 9. As can be seen, MO-LNS consistently obtains better or equivalent results compared to the MO-CP approach. The hypervolume values reached by MO-LNS are very close to the optimal ones.

Table 2. Comparison of MO-CP and MO-LNS on standard instances of the Bi-Objective Knapsack Problem.

			$\mathcal{H}_S(10^8)$		$\|S\|$		$\|S \cap S^*\|$	
Instance	$\|S^*\|$	$\mathcal{H}_{S^*}(10^8)$	MO-LNS	MO-CP	MO-LNS	MO-CP	MO-LNS	MO-CP
2KP100A	172	15,59	**15,59**	15,05	**172**	128	**172**	112
2KP100B	174	15,12	**15,12**	14,62	**170,6**	124	**164,7**	93
2KP100C	64	16,68	**16,68**	**16,68**	**64**	**64**	**64**	**64**
2KP100D	76	16.31	**16.31**	16.28	**76**	73	**76**	73
2KP150A	244	39.66	**39.66**	35.42	**226.2**	88	**187**	31
2KP150B	348	41.46	**41.46**	36.31	**303.3**	91	**192.4**	51
2KP150C	166	34.17	**34.17**	33.15	**155.6**	83	**127**	34
2KP150D	207	36.04	**36.04**	32.88	**199.8**	88	**155.6**	62
2KP200A	439	64.34	**64.34**	57.43	**361**	86	**178.2**	34
2KP200B	397	65.78	**65.77**	58.82	**345.2**	108	**232.8**	54
2KP200C	328	57.48	**57.46**	48.30	**297.8**	60	**187.1**	18
2KP200D	361	73.42	**73.40**	62.71	**304.1**	64	**176.9**	29
2KP250A	629	94.37	**94.34**	78.68	**433.5**	70	**95.5**	12
2KP250B	629	89.67	**89.65**	74.81	**410.9**	90	**107.9**	44
2KP250C	528	91.25	**91.24**	75.30	**383.3**	72	**108.9**	22
2KP250D	424	66.56	**66.55**	56.98	**303.4**	51	**142.8**	26

5.3 Tank Allocation Problem

The tank allocation problem involves the assignment of different cargoes (volumes of chemical products to be shipped by the vessel) to the available tanks of the vessel [24]

[9] The optimal fronts were provided by the creator of MOCOLib [28]

while satisfying hard segregation constraints *e.g.* to avoid placing dangerous cargoes in adjacent tanks. An ideal loading plan should maximize the total volume of unused tanks (*i.e.* free space) to minimize cleaning costs (objective 1). Minimizing the number of used tanks (objective 2) is also desirable in order to maximize the chances of accommodating other cargoes in next visited ports. The LNS model used in our experiment is the same as the one introduced in [24] except that it is now bi-objective. The MO-LNS parameters are:

- The relaxed solution is chosen randomly from the current archive.[10]
- Chose randomly to diversify or intensify with equal probability.

Results are given in Table 5.3. The columns have the same meaning as in previous result tables. The exact Pareto fronts were generated with the van Wassenhove and Gelders Algorithm using a MIP solver (Gurobi 5.02) that took about 3 minutes to run on each instance. The MO-LNS framework finds the exact Pareto front of each instance within a timeout of 60 seconds. We also indicate for the same model the results obtained for a MO-CP approach with a timeout of 300 seconds. This MO-CP approach is only able to discover one solution of the exact Pareto front of the first instance (chemicalA). The hypervolume indicator shows that other nondominated solutions discovered with MO-CP remain quite far from optimal ones.

Table 3. Comparison of MO-CP and MO-LNS on real-life instances of the Bi-Objective Tank Allocation Problem. MO-LNS finds all the exact Pareto fronts.

			$\mathcal{H}_S(10^8)$		$\|S\|$		$\|S \cap S^*\|$	
Instance	$\|S^*\|$	$\mathcal{H}_{S^*}(10^8)$	MO-LNS	MO-CP	MO-LNS	MO-CP	MO-LNS	MO-CP
chemicalA	6	1848	**1848**	1022	**6**	4	**6**	1
chemicalB	7	2976	**2976**	1010	**7**	3	**7**	0
chemicalC	8	3597	**3597**	852	**8**	2	**8**	0
chemicalD	8	5358	**5358**	555	**8**	1	**8**	0

6 Future works and Conclusion

This paper introduced the MO-LNS framework; an extension of LNS to efficiently solve MOCO problems with CP. MO-LNS uses the `Pareto` constraint to maintain a best-so-far archive that is iteratively improved by diversification and intensification. Modeling abstractions were presented into the OscaR solver to select at each restart a solution in the archive, and to diversify or intensify the front MO-LNS was experimented on various MOCO problems showing its superiority over MO-CP to get close to optimal hypervolumes. The Scala source-code of our implementation as well as some complete MO-LNS examples are available on OscaR repository [20].

As future work, we plan to study adaptive diversification/intensification strategies. We would like to explore the parallelization of MO-LNS. Finally we want to tackle more complex problems with MO-LNS, such as multi-objective scheduling or vehicle routing problems.

[10] No real impact since the optimal front is very small.

References

1. Ajith Abraham and Lakhmi Jain. Evolutionary multiobjective optimization. *Evolutionary Multiobjective Optimization*, 2005.
2. J. Christopher Beck. Solution-guided multi-point constructive search for job shop scheduling. *J. Artif. Int. Res.*, 29(1):49–77, May 2007.
3. R. Bent and P.V. Hentenryck. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research*, 33(4):875–893, 2006.
4. S. Cahon, N. Melab, and E.-G. Talbi. Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. *Journal of Heuristics*, 10(3):357–380, May 2004.
5. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-2. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.
6. Juan J. Durillo and Antonio J. Nebro. jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42:760–771, 2011.
7. Matthias Ehrgott. *Multicriteria optimization*, volume 2. Springer Berlin, 2005.
8. Matthias Ehrgott and Xavier Gandibleux. Hybrid metaheuristics for multi-objective combinatorial optimization. *Hybrid metaheuristics*, pages 221–259, 2008.
9. M. Gavanelli. An algorithm for multi-criteria optimization in csps. *ECAI*, 2:136–140, 2002.
10. Yacov Y Haimes, Leon S Lasdon, and David A Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 1(3):296–297, 1971.
11. Pascal Van Hentenryck and Laurent Michel. Nondeterministic control for hybrid search. *Constraints*, 11(4):353–373, 2006.
12. Joshua Knowles. Towards landscape analyses to inform the design of a hybrid local search for the multiobjective quadratic assignment problem. *Soft computing systems: design, management and applications*, 2002:271–279, 2002.
13. Joshua Knowles and David Corne. Instance generators and test suites for the multiobjective quadratic assignment problem. In *Second International Conference on Evolutionary Multi-Criterion Optimization EMO 2003*, number 2632 in LNCS, pages 295–310. Springer, 2003. Available from `http://www.cs.man.ac.uk/ jknowles/mQAP/`.
14. P. Laborie and D. Godard. Self-adapting large neighborhood search: Application to single-mode scheduling problems. *Proceedings MISTA-07, Paris*, pages 276–284, 2007.
15. Jean-Baptiste Mairy, Yves Deville, and Pascal Van Hentenryck. Reinforced adaptive large neighborhood search. In *8th Workshop on Local Search techniques in Constraint Satisfaction (LSCS 2011). A Satellite Workshop of CP*, Perugia, Italy, 2011.
16. Jean-Baptiste Mairy, Pierre Schaus, and Yves Deville. Generic adaptive heuristics for large neighborhood search. In *Seventh International Workshop on Local Search Techniques in Constraint Satisfaction (LSCS2010). A Satellite Workshop of CP*, 2010.
17. R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.
18. D. Mehta, B. OSullivan, and H. Simonis. Comparing solution methods for the machine reassignment problem. In *Principles and Practice of Constraint Programming*, pages 782–797. Springer, 2012.
19. Laurent Michel, Alexander Shvartsman, Eldine Sonderegger, and Pascal Van Hentenryck. Optimal deployment of eventually-serializable data services. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 188–202, 2008.

20. OscaR Team. OscaR: Scala in OR, 2012. Available from `https://bitbucket.org/oscarlib/oscar`.
21. D. Pacino and P. Van Hentenryck. Large neighborhood search and adaptive randomized decompositions for flexible jobshop scheduling. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 1997–2002. AAAI Press, 2011.
22. Laurent Perron, Paul Shaw, and Vincent Furnon. Propagation guided large neighborhood search. *Principles and Practice of Constraint Programming–CP 2004*, pages 468–481, 2004.
23. P. Schaus, P. Van Hentenryck, J.N. Monette, C. Coffrin, L. Michel, and Y. Deville. Solving steel mill slab problems with constraint-based techniques: Cp, lns, and cbls. *Constraints*, 16(2):125–147, 2011.
24. Pierre Schaus, Jean-Charles Régin, Rowan Van Schaeren, Wout Dullaert, and Birger Raa. Cardinality reasoning for bin-packing constraint: application to a tank allocation problem. In *Principles and Practice of Constraint Programming*, pages 815–822. Springer, 2012.
25. P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. *Principles and Practice of Constraint ProgrammingCP98*, pages 417–431, 1998.
26. EL Ulungu and J Teghem. Multi-objective combinatorial optimization problems: A survey. *Journal of Multi-Criteria Decision Analysis*, 3(2):83–104, 1994.
27. Luc N. Van Wassenhove and Ludo F. Gelders. Solving a bicriterion scheduling problem. *European Journal of Operations Research*, 4:42–48, 1980.
28. Xavier Gandibleux. A collection of test instances for multiobjective combinatorial optimization problems, 2013. Available from `http://xgandibleux.free.fr/MOCOlib/`.
29. Eckart Zitzler, Marco Laumanns, Lothar Thiele, Eckart Zitzler, Eckart Zitzler, Lothar Thiele, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm, 2001.
30. Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and V Grunert da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *Evolutionary Computation, IEEE Transactions on*, 7(2):117–132, 2003.