5 Implementierung eines Test-Frameworks

Auf Grundlage der vorgestellten Komponentenstruktur der ACO-Metaheuristik des Kapitels 3.3 wird nachfolgend eine Anwendungsstruktur benötigt, die einerseits einfache Erweiterung um neue Komponentenausprägungen und ihre Wiederverwendung ermöglicht. Zum anderen soll die Komposition von ACO-Algorithmen zwecks ihrer Evaluation durch Auswahl der Komponenten möglichst einheitlich und automatisiert möglich sein.

5.1 Abstrakte Komponenten

Der praktischen Anwendung von Metaheuristiken geht die problemspezifische Anpassung und Implementierung voraus. Der hierfür entstehende Aufwand ist der Grund dafür, dass die Metaheuristiken in der Praxis nur selten eingesetzt werden (Fink und Voß 2003, S. 395). Frameworks stellen Artefakte der Wirtschaftsinformatik dar und erleichtern schnelle Erstellung lauffähiger Prototypen. Der Kerngedanke dieses Kapitels ist, generalisierte Komponenten der ACO-Metaheuristiken als ein Programmiergerüst bereitzustellen, um die Erweiterung und Wiederverwendung für die Komposition von ACO-Algorithmen zu ermöglichen.

Das objektorientierte Paradigma deckt sich weitgehend mit der Formulierung der ACO-Metaheuristik als eine Komponentenstruktur. Die Komponentenstruktur des Kapitels 3.2 ist auf der generalisierten Ebene formuliert und gibt dort die Abhängigkeiten zwischen den Komponenten vor, welche der ACO-Metaheuristik eigen sind. Die generalisierten Komponenten können durch die Konzepte der Kapselung und Vererbung abgebildet werden. Durch Polymorphie können automatisiert spezialisierte Komponenten in Übereinstimmung mit den Abhängigkeiten der generalisierten Ebene miteinander kombiniert werden.

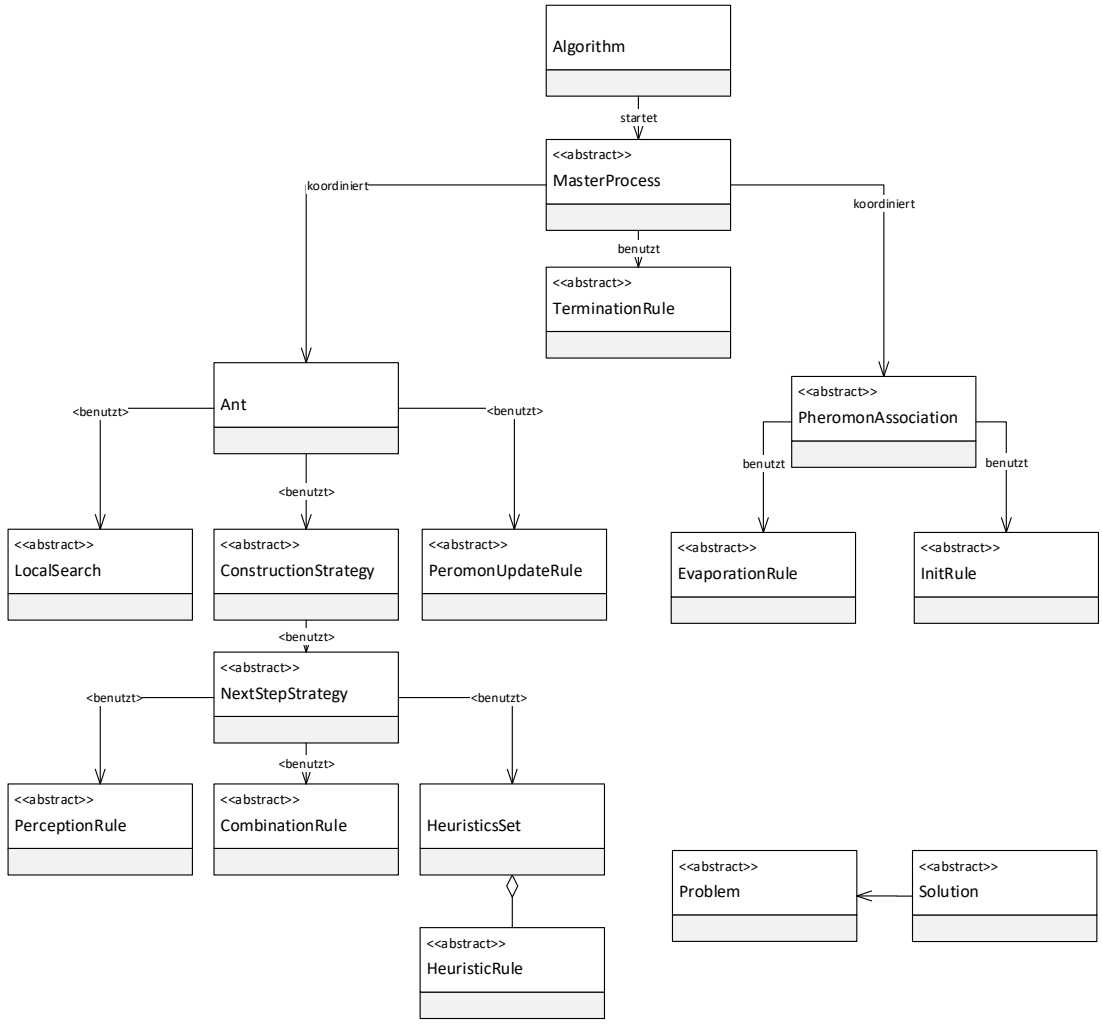

Abbildung 5.1-1: UML-Diagramm des ACO-Frameworks

In der Abbildung 5.1-1 ist das UML-Klassendiagramm des abstrakten Frameworks dargestellt, welches noch keine Spezialisierung an ein bestimmtes Problem aufweiset. Die Klassen entsprechen im Einzelnen weitestgehend den generalisierten Komponenten der ACO-Metaheuristik und sind selbsterklärend. Durch die Schnittstellen sind die Abhängigkeiten zwischen den Komponenten auf der abstrakten Ebene vorgegeben. Eine wesentliche Änderung betrifft die Kapselung der Komponenten *LocalSearch*, *ConstructionStrategy* und *PheromonUpdateRule* in der Klasse *Ant*. Auch kapselt die Klasse *PheromoneAssociation* die Funktionalität der Pheromon-Evaporation und -Initiierung.

5.2 Spezialisierung der Komponenten

Für die Nutzung des Frameworks zur Anpassung an ein bestimmtes Problem ist die Ableitung aller abstrakten Klassen und Implementierung der dort vorhandenen Schnittstellen notwendig. Durch die Ableitung werden die Komponenten mit der Anwendungslogik

ausgestattet. Die Abbildung 5.2-1 zeigt beispielhaft die Spezialisierung der abstrakten Klasse *MasterProcess* durch die Klasse *MasterProcessElitist*.

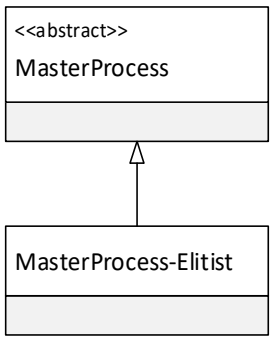

Abbildung 5.2-1: UML-Diagramm der Vererbung am Beispiel des Masterprozesses

Die in den Konfigurationen von ACO-Algorithmen des Kapitels 4 verwendeten Komponenten des Kapitels 3.3 wurden entsprechend diesem Vorgehen implementiert.

5.3 Konfiguration und Ausführung von ACO-Algorithmen

Die Konfiguration der ACO-Algorithmen erfolgt mittels des Setzens benötigter Referenzen über Klassen-Konstruktoren (Constructor Injection). Die instanziierten Komponenten speichern die Referenzen als Objektvariablen und erzeugen insgesamt eine der Abbildung 5.1-1 entsprechende spezialisierte lauffähige Struktur eines ACO-Algorithmus ab.

Aufgrund dessen, dass die es sich bei ACO um eine populationsbasierte Metaheuristik handelt, kann diese relativ leicht auf der Ebene der Individuen weitgehend parallel ausgeführt werden (Dorigo und Stützle 2010, S. 252). Die Ausführung der konfigurierten Algorithmen konnte vor allem bei großen SCP-Instanzen beschleunigt werden. Dazu wurden mehrere Schritte im Masterprozess parallelisiert, die keinen Austausch von Informationen auf kollektiver Ebene der Ameisen erfordern, um die Threadsicherheit zu garantieren.

Über die Kernkomponenten der ACO-Metaheuristik hinaus wurden Techniken für die Verwaltung von ACO-Komponenten-Konfigurationen, Bereitstellung der Benchmark-Instanzen, graphische und tabellarische Darstellung benötigt. Diese Teile des Programms wurden in eigenständige Pakete unterteilt und werden hier nicht weiter beschrieben.